REMARKS

I. General

Claims 1, 2, 4-24, and 27-31 were pending in the present application, and all of the pending claims are rejected in the current Final Office Action (mailed March 11, 2005). The outstanding issues raised in the Final Office Action are:

- Claims 1, 2, 4-24, and 27-31 are rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,872,971 issued to Knapman et al (hereinafter "*Knapman*") in view of U.S. Patent No. 6,526,416 issued to Long (hereinafter "*Long*").

In response, Applicant respectfully traverses the outstanding claim rejections, and requests reconsideration and withdrawal thereof in light of the remarks presented herein.

II. Rejection under 35 U.S.C. § 103(a)

To establish a prima facie case of obviousness, three basic criteria must be met. *See* M.P.E.P. § 2143. First, there must be some suggestion or motivation, either in the applied references themselves or in the knowledge generally available to one of ordinary skill in the art, to combine the reference teachings. Second, there must be a reasonable expectation of success. Finally, the applied references must teach or suggest all the claim limitations. Without conceding any other criteria, Applicant respectfully asserts that the applied combination of *Knapman* and *Long* does not teach or suggest all the claim limitations.

A. Independent Claim 1

Independent claim 1 recites, in part, "one or more non-transactional resources; at least one component that defines one or more tasks executable by at least one of said one or more non-transactional resources". The current Office Action asserts that *Knapman* teaches these elements of claim 1, *see* item 4 on page 2 and item 36 on page 8 of the current Office Action. Applicant respectfully disagrees, as discussed below.

As Applicant pointed out in the Amendment of November 12, 2004, *Knapman* fails to address non-transactional resources. Rather, *Knapman* teaches a system that enables "operation requests from an application program which implements one transactional model

to be translated to operations which are meaningful to a transaction processing manager which implements a different transactional model." Col. 5, lines 29-34. *Knapman* recognizes that different transactional models exist, such as the CICS transactional model and the OTS transactional model, *see* Col. 3, lines 13-19. The operational mismatch between the different transactional models "currently prevents interoperation between data processing resources which implement the different models." Col. 3, lines 20-22.

In response to this argument, the Final Office Action asserts that *Knapman* "teaches both transactional and nontransactional information and how each can be distinctly marked (col. 2 lines 51-55 col. 9 lines 40-42)", *see* item 36 on page 8 of Final Office Action. The cited portions of *Knapman* describe an OTS transactional system, which provides an unchained model, and a CICS transactional system that implements a chained model. For instance, column 2, lines 26-55 of *Knapman* provides:

> A number of different transaction processing models are known in the art. Some transaction processing systems require explicit transactional demarcation--i.e. an application program must issue the relevant API commands for operations to begin, commit or rollback a transaction. This requirement is a feature of the Object Management Group, Inc.'s (OMG's) Object Transaction Service (OTS) specification--a specification for a proposed service which supports transactional behaviour in a distributed heterogeneous environment based on the OMG Common Object Request Broker Architecture (CORBA). CORBA defines the interactions between objects of an object oriented system (see below), and in particular between distributed objects in different systems. The OTS exploits object oriented programming to encapsulate the processing performed under transactional scope, allowing a programmer (or similar person) to designate certain classes of operations as transactional. The unchained model of transactions is supported since a client program can explicitly start, suspend and resume transactions. Invoking a transactional operation outside a transaction causes the operation to be performed outside the scope of any transaction. In the unchained model, a program can explicitly enter and leave the scope of transactions so that some of its execution might be within the scope of one transaction, some within the scope of another, and some outside the scope of any transaction. Unlike the non-object-oriented unchained model, in which a program can freely mix transactional and non-transactional work, OTS requires that the transactional operations be distinctly marked.

Thus, this portion of *Knapman* describes that different transactional processing models are known. It further describes that OTS is one transactional processing system which provides an unchained model in which a program can explicitly enter and leave the

scope of transactions so that some of its execution might be within the scope of one transaction, some within the scope of another, and some outside the scope of any transaction. To support such mixing of transactional and non-transactional work, OTS requires that the transactional operations be distinctly marked within a program.

Thus, OTS provides a transactional processing system. While operations of a program may be performed outside the scope of a transaction, OTS provides transactional resources that are capable of performing transactional tasks when so desired. That is, if a transaction is desired, the transactional resources of OTS are used for achieving such transaction. Thus, OTS provides transactional resources that can be used when tasks are desired to be performed as a transaction.

At column 2, line 60 through column 3, line 7, *Knapman* provides:

> The CICS on-line transaction processing programs, which are commercially available from International Business Machines Corporation, support "chained" transactions across cooperating systems. Chained transactions have the property that a program (e.g. a CICS application) is always within the scope of some transaction, and a system that implements the chained model can maintain the rule that all transactional operations must be executed under its control (by means of transaction programs in the case of CICS systems). A CICS transaction begun in one system may include operations performed by other systems, with demarcation of the transaction (e.g. commit processing) being controlled by a CICS system without requiring application-initiated demarcation operations.

Thus, CICS is a transactional processing system that supports chained transactions wherein a program is always within the scope of some transaction, as opposed to OTS where the program can enter and leave transactions. Thus, CICS also provides transactional resources. *Knapman* further describes at column 3, lines 13-22:

> It is an important distinction between the CICS transactional model and the OTS transactional model that CICS servers manage units of work and support chained transactions whereas OTS requires client application control and implements an unchained transaction model, since the respective transactions look very different, and require different action, from the client application. <u>This mismatch between the transactional models currently prevents interoperation between data processing resources which implement the different models</u>. (Emphasis added).

Thus, *Knapman* is concerned with the mismatch between the CICS and OTS transactional models. For instance, at column 8, lines 49-55 *Knapman* states:

> While the invention relates generally to achieving interoperability between data processing resources which implement different transactional models, the invention is particularly useful in providing a solution to the mismatch between resources which support chained transactions, such as CICS transaction processing systems, and the unchained transactional model implemented in OTS. (Emphasis added).

Knapman describes operation of CICS transactions as follows:

> CICS transactions are generally begun by an end user signing on to a CICS system and then invoking a particular application that they intend to use, generally by typing the transaction identification code (id) or using a predefined program function key. CICS looks up the transaction identifier in its internal Program Control Table where it finds out which program to invoke first to execute the requested transaction. Subsequent delimiting of the transaction is controlled by the CICS system's implicit management of units of work; CICS coordinates updates when a syncpoint request is received.
>
> Client-server programming in CICS uses two application programming interfaces (APIs) that provide external access to CICS facilities, allowing non-CICS applications to gain access to CICS facilities and data:
>
> External call interface (ECI)
> External presentation interface (EPI)
>
> FIG. 2 illustrates the prior art use of the external interfaces 100 by a non-CICS application 110 which is located in a client system 120 and is using the facilities of CICS 130 in a server system 140. The CICS client software 150 processes the application's EPI and ECI requests, and transmits them to the server system using an appropriate communication protocol. Although FIG. 2 shows separate client and server systems, the configuration may be on a single workstation. The installation of a CICS client program on a data processing system enables that system to connect to an appropriate CICS server program. (Col. 8, line 65 – Col. 9, line 25)...
>
> The ECI allows a non-CICS application to call a CICS program in a CICS server and to initiate a CICS transaction. The application can be connected to several servers at the same time, and it can have several program calls outstanding at the same time. The CICS program can access and update all CICS resources except performing terminal input/output. (Col. 9, lines 41-46)...
>
> As compared with the basic model of chained transactions (where a program is always within the scope of a transaction), the CICS ECI affords greater

> flexibility by extending the concept of a chained transaction with the concept of a "mirror transaction". The CICS ECI allows a client program to execute outside the scope of a transaction but to invoke several CICS application programs under the scope of one or more transactions. The client program identifies these transactions by means of a logical unit of work identifier. An invocation of an application program (when the task implemented by that application program is selected to be performed) is termed a transaction scheduling. Much of the simplicity of the chained transactional model is retained.
>
> The transaction programs, however, must always be separate from the client program, these being executed under the control of a server program. The transactional programs are always within the scope of a transaction as a consequence of the chained transaction model. (Emphasis added). (Col. 9, line 64 – col. 10, line 14).

In view of the above, *Knapman* does not address using non-transactional resources for performing a transaction. Rather, *Knapman* addresses use of transactional resources, which may be invoked by an application program to perform a desired transaction. For instance, a non-CICS application may interface with a CICS application to invoke a transaction to be performed via the CICS system's transactional resources. If a transaction is desired in *Knapman*, transactional resources are used by invoking a transactional processing system, such as CICS, rather than using non-transactional resources as recited by claim 1.

Knapman further addresses a technique for translating from a first transactional model to a different transactional model to enable interoperation of transactional resources of each model (e.g., OTS transactional resources and CICS transactional resources), but *Knapman* fails to teach or suggest non-transactional resources in the manner recited in claim 1.

Thus, because the current Office Action relies on *Knapman* as teaching or suggesting the above elements of claim 1, the present rejection is improper as the Office Action has failed to establish a prima facie case of obviousness.

Although the present Office Action does not rely upon *Long* as teaching this element, *Long* appears to address utilizing non-transactional resources in performing transactional operations, *see e.g.,* Col. 1, lines 5-10; col. 2, lines 60-62; col. 3, lines 2-6; and col. 10, lines 55-61. However, claim 1 recites "resource manager operable to control execution of said one or more tasks defined by said at least one component as a transaction for activation of a

service" (emphasis added). *Long* fails to teach or suggest controlling execution of defined tasks by non-transactional resources for activating a service.

In response to this argument raised in Applicant's amendment of November 12, 2004, the Final Office Action asserts that *Long* "teaches a resource manager able to activate a transaction for service (col. 10 lines 35-44; col. 27 lines 4-27)", *see* item 36 on page 9 of Final Office Action. While *Long* proposes a resource manager that supports performing transactions, *Long* does not teach or suggest activating a service as a transaction. The Final Office Action notes that *Long*'s resource manager is able to "activate a transaction". However, this is not what claim 1 recites. Rather, claim 1 recites "resource manager operable to control execution of said one or more tasks defined by said at least one component as a transaction for activation of a service" (emphasis added). Again, while the resource manager of *Long* supports transactions, it does not teach or suggest performing one or more tasks as a transaction for activation of a service, such as activation of a web hosting service for hosting a new web site for a customer (*see e.g.*, page 2, line 1 – page 3, line 7 of the present application).

In view of the above, the applied combination of *Knapman* and *Long* fails to teach or suggest all elements of claim 1, and thus claim 1 is not obvious under 35 U.S.C. § 103(a) over this combination.

B. Independent Claim 15

Independent claim 15 recites, in part, "at least one component defining one or more tasks executable by at least one of said one or more non-transactional resources". The current Office Action rejects claim 15 "based on the same rejection for claim 1". Thus, as discussed above with claim 1, the Office Action relies on *Knapman* as teaching the above element. As further discussed above with claim 1, *Knapman* fails to teach or suggest one or more non-transactional resources, and thus also fails to teach or suggest at least one component defining one or more tasks executable by at least one of the non-transactional resource(s).

Thus, because the current Office Action relies on *Knapman* as teaching or suggesting the above element of claim 15, the present rejection is improper as the Office Action has failed to establish a prima facie case of obviousness.

While *Long* appears to address utilizing non-transactional resources in performing transactional operations, *see e.g.,* Col. 1, lines 5-10; col. 2, lines 60-62; col. 3, lines 2-6; and col. 10, lines 55-61, *Long* fails to teach or suggest "said resource manager controlling execution of said at least one component to perform said plurality of tasks as a transaction for service provisioning" (emphasis added), as recited by claim 15. While *Long* teaches a resource manager, *Long* does not teach controlling execution of a plurality of tasks by a non-transactional resource as a transaction "for service provisioning".

In view of the above, the applied combination of *Knapman* and *Long* fails to teach or suggest all elements of claim 15, and thus claim 15 is not obvious under 35 U.S.C. § 103(a) over this combination.

C. Independent Claim 23

Independent claim 23 recites, in part, "code for controlling one or more non-transactional resources to perform said plurality of tasks as a transaction, wherein said code for controlling one or more non-transactional resources includes code for invoking performance of a task by said one or more non-transactional resources, and wherein said code for invoking performance of a task includes code for calling a function defined by a plugin component that is communicatively coupled to said one or more non-transactional resources" (emphasis added). The combination of *Knapman* and *Long* fails to teach or suggest at least the above element of claim 23. As described with claim 1 above, *Knapman* does not address non-transactional resources. While *Long* addresses non-transactional resources, *Long* fails to teach or suggest calling a function defined by a plugin component that is communicatively coupled to the non-transactional resources for invoking performance of a task by the non-transactional resources.

The Final Office Action asserts that *Knapman* teaches "wherein said code for invoking performance of a task includes code for calling a function defined by a plugin component that is communicatively coupled to said one or more non-transactional resources" citing to col. 10, lines 1-5, col. 5, lines 19-27, and col. 6, lines 58-62 of *Kapman*, *see* item 28 on page 7 of Final Office Action. Col. 10, lines 1-5 of *Knapman* provides:

> The CICS ECI allows a client program to execute outside the scope of a transaction but to invoke several CICS application programs under the scope of one or more transactions. The client program identifies these transactions by means of a logical unit of work identifier.

This does not teach or suggest code for calling a function defined by a plugin component that is communicatively coupled to said one or more non-transactional resources. Rather, the ECI is an interface for the CICS transactional processing system, which provides transactional resources, rather than non-transactional resources.

Col. 5, lines 10-27 of *Knapman* provides:

> According to a first aspect of the present invention, there is provided a data processing system including data processing resources which implement a first transactional model, in which all transactional operations are performed within transactional scope, and means for interfacing between said resources and data processing resources which implement a second transactional model, in which transactional operations may be invoked and performed outside of transactional scope, thereby to enable interoperation between said different resources in the processing of a transaction, the means for interfacing including: means for mapping between transactional operations according to the second model and transactional operations according to the first model, said mapping including designating transactional operations of said second model which are invoked outside of a first transaction as separate transactions according to the first model which are isolated from said first transaction; and means for maintaining a list of said operation mappings for the transaction.

This does not teach or suggest code for calling a function defined by a plugin component that is communicatively coupled to said one or more non-transactional resources. Rather, this proposes an interface between a first transactional model and a second transactional model. Such transactional models each use transactional resources rather than non-transactional resources.

Col. 6, lines 56-65 of *Kapman* provides:

> The external interface of the transaction processing program may be an external interface of a client program of a client-server transaction service. The isolated transactions will possess all of the properties required of a non-transactional operation, whereas the additional properties that they can be rolled back or committed may be invisible and unavailable to the client program. However, the client can rely on marking and encapsulation according to the second model to facilitate identification of the transactional operations and their performance by a server program.

This also does not teach or suggest code for calling a function defined by a plugin component that is communicatively coupled to said one or more non-transactional resources. No plugin that is coupled to a non-transactional resource is mentioned. Further, this proposes an external interface of a client program of a transaction service that enables interoperation between two different transactional models. For instance, it allows for marking to identify transactional operations to be performed by transactional resources of a transactional model. Thus, if an operation is identified as transactional, the transactional resources of a transactional model (e.g., the CICS system or OTS system) can be used for processing such transaction.

Again, *Knapman* does not teach or suggest using non-transactional resources for performing a transaction, and fails to teach or suggest code for calling a function defined by a plugin component that is communicatively coupled to said one or more non-transactional resources.

In view of the above, the applied combination of *Knapman* and *Long* fails to teach or suggest all elements of claim 23, and thus claim 23 is not obvious under 35 U.S.C. § 103(a) over this combination.

D. Independent Claim 27

Claim 27 recites:

> A system comprising:
> non-transactional resources for providing a plurality of different services;
> a plugin for each of said plurality of different services that defines one or more tasks for activating the respective service; and
> resource manager operable to control execution of said plugins to selectively activate multiple ones of the plurality of different services as a transaction. (Emphasis added).

The combination of *Knapman* and *Long* fails to teach or suggest each of the above elements of claim 27. For instance, *Knapman* does not teach or suggest non-transactional resources for providing a plurality of different services and using a resource manager to control execution of plugins to selectively activate multiple ones of the plurality of different services as a transaction. Rather, if an operation is desired to be performed as a transaction in

Knapman, transactional resources of a transactional model are used for performing such transaction. *Knapman* focuses on enabling use of such transactional resources of a plurality of different types of transactional models, such as OTS and CICS. *Knapman* does not teach or suggest performing any operations, including those for activating services, as a transaction using non-transactional resources.

As described above, *Long* appears to propose use of non-transactional resources. However, *Long* fails to teach or suggest a "resource manager operable to control execution of said plugins to selectively activate multiple ones of the plurality of different services as a transaction" (emphasis added). *Long* does not teach or suggest activating different services as a transaction, such as activating ones of "web hosting service, database service, software application service, DNS service, LDAP service, monitoring service, management service, quality of service (QoS) service, usage measurement service, and billing service", as further recited in claim 28.

In view of the above, the applied combination of *Knapman* and *Long* fails to teach or suggest all elements of claim 27, and thus claim 27 is not obvious under 35 U.S.C. § 103(a) over this combination.

E. Independent Claim 29

Claim 29 recites:

> A system comprising:
> at least one non-transactional resource for providing at least one service associated with a web hosting service;
> a plugin for said at least one non-transactional resource that defines one or more tasks for activating the at least one associated service for a given web hosting service; and
> resource manager operable, when activating said given web hosting service, to control execution of said plugin to activate said at least one associated service as a transaction with activating said given web hosting service. (Emphasis added).

The combination of *Knapman* and *Long* fails to teach or suggest each of the above elements of claim 29. For instance, *Knapman* does not teach or suggest at least one non-transactional resources for providing at least one service associated with a web hosting service and using a resource manager to control execution of plugins to activate the at least

one associated service as a transaction when activating a given web hosting service. Rather, if an operation is desired to be performed as a transaction in *Knapman*, transactional resources of a transactional model are used for performing such transaction. *Knapman* focuses on enabling use of such transactional resources of a plurality of different types of transactional models, such as OTS and CICS. *Knapman* does not teach or suggest performing any operations, including those for activating services associated with a web hosting service, as a transaction using non-transactional resources.

As described above, *Long* appears to propose use of non-transactional resources. However, *Long* fails to teach or suggest a "resource manager operable, when activating said given web hosting service, to control execution of said plugin to activate said at least one associated service as a transaction with activating said given web hosting service." *Long* does not teach or suggest activating at least one associated service as a transaction when activating a web hosting service, such as activating ones of "database service, software application service, DNS service, LDAP service, monitoring service, management service, quality of service (QoS) service, usage measurement service, and billing service", as further recited in claim 31.

In view of the above, the applied combination of *Knapman* and *Long* fails to teach or suggest all elements of claim 29, and thus claim 29 is not obvious under 35 U.S.C. § 103(a) over this combination.

F. Dependent Claims 2, 4-14, 16-22, 24, 28, and 30-31

Dependent claims 2, 4-14, 16-22, 24, 28, and 30-31 stand rejected as unpatentable under 35 U.S.C. § 103(a) over the combination of *Knapman* and *Long*. In view of the above, Applicant respectfully submits that independent claims 1, 15, 23, 27, and 29 are not unpatentable under 35 U.S.C. § 103(a) over the combination of *Knapman* and *Long*. Further, each of dependent claims 2, 4-14, 16-22, 24, 28, and 30-31 depend either directly or indirectly from one of independent claims 1, 15, 23, 27, and 29 and thus inherit all limitations of the respective independent claim from which they depend. It is respectfully submitted that dependent claims 2, 4-14, 16-22, 24, 28, and 30-31 are allowable not only because of their dependency from their respective independent claims for the reasons discussed above, but also in view of their novel claim features (which both narrow the scope of the particular

claims and compel a broader interpretation of the respective base claim from which they depend).

III. Conclusion

In view of the above, Applicant believes the pending application is in condition for allowance.

Applicant believes no fee is due with this response. However, if a fee is due, please charge our Deposit Account No. 08-2025, under Order No. 10012815-1 from which the undersigned is authorized to draw.

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Label No. EV 482708474US in an envelope addressed to: M/S After Final, Commissioner for Patents, Alexandria, VA 22313.

Date of Deposit: April 27, 2005

Typed Name: Gail L. Miller

Signature: Gail L. Miller

Respectfully submitted,

By: _____

Jody C. Bishop
Attorney/Agent for Applicant(s)
Reg. No. 44,034
Date: April 27, 2005
Telephone No. (214) 855-8007